AD-A051 383    FOREIGN TECHNOLOGY DIV WRIGHT-PATTERSON AFB OHIO    F/G 9/2
CONSTRUCTION OF MINIMAL-REDUNDANCY CODES FOR VARIABLE-LENGTH WO--ETC(U)
AUG 77  B HORVAT

UNCLASSIFIED    FTD-ID(RS)T-1312-77    NL

| OF |

AD
A051 383

END
DATE
FILMED
4-78
DDC

FTD-ID(RS)T-1312-77

1

# FOREIGN TECHNOLOGY DIVISION

## CONSTRUCTION OF MINIMAL-REDUNDANCY CODES FOR VARIABLE-LENGTH WORDS

by

B. Horvat

Approved for public release;
distribution unlimited.

# EDITED TRANSLATION

FTD-ID(RS)T-1312-77 9 August 1977

MICROFICHE NR: FTD-77-C-001019

CONSTRUCTION OF MINIMAL-REDUNDANCY CODES FOR VARIABLE-LENGTH WORDS

By: B. Horvat

English pages: 10

Source: Automatika, Vol. 17, No. 1-2, 1976, PP. 43-46

Country of origin: Yugoslavia
Translated by: LINGUISTIC SYSTEMS, INC
F33657-76-D-0389
Oljan Repic
Requester: FTD/ETCK
Approved for public release; distribution unlimited

| THIS TRANSLATION IS A RENDITION OF THE ORIGINAL FOREIGN TEXT WITHOUT ANY ANALYTICAL OR EDITORIAL COMMENT. STATEMENTS OR THEORIES ADVOCATED OR IMPLIED ARE THOSE OF THE SOURCE AND DO NOT NECESSARILY REFLECT THE POSITION OR OPINION OF THE FOREIGN TECHNOLOGY DIVISION. | PREPARED BY:<br>TRANSLATION DIVISION<br>FOREIGN TECHNOLOGY DIVISION<br>WP-AFB, OHIO. |
|---|---|

# CONSTRUCTION OF MINIMAL-REDUNDANCY CODES FOR VARIABLE-LENGTH WORDS

Bogomir Horvat, M.S.

Another approach for the construction of minimum-redundancy codes for variable-length words is described. Based on Huffman's optimized codes, it uses a small memory space and a shorter program. The word-length is inversely proportional to the probability of its presence at the output. The construction of minimum-redundancy codes is not limited to data transfer but can also be used in the memory, for display of system states, for language generation, etc. The algorithm is computer tested. Two illustrative examples are given.

## INTRODUCTION

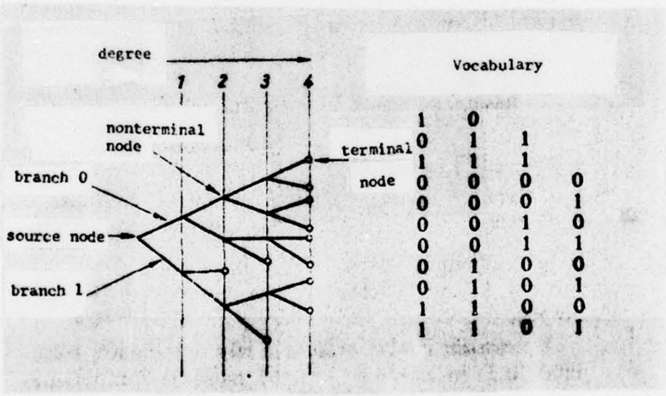
Let us look at a data source without memory capability that generates discrete symbols from the series $X = (x_1, x_2, x_3, \ldots, x_M)$ in constant time intervals. The appearance frequency of individual symbols is determined by the probability $P(x_i) = (1/M)$, $i = 1,2,\ldots,M$. Every probability value is constant as the data system has no memory capability. Let us assume that all the allowed mistakes and errors that perhaps influence the transmission are transposed into the data source. Using the two simplifications one avoids
1. mathematical complications due to statistical interdependence,
2. one is dealing with a noiseless communication channel during the data transmission.

The discrete signals are all valued equally during data processing, and their presence in time or their transmission costs the same. We will concentrate on coding symbol series from M-finite alphabet into words consisting of L symbols of ($M \geqq L = 2$)-finite alphabet. We are after new variable-length codewords which can be uniquely and instantly decoded and the average length of which is minimal. An optimal code has been sought by Shannon and Fano. The approach and the characteristics of that optimal code have been described by Huffman[1]. The disadvantage of Huffman's algorithm is the fact that it requires a rather large memory, and that the entire symbol and statistics index table must be maintained throughout the procedure.

As is known, code words of length $n_1, n_2, \ldots, n_M$ fulfill the condition if they satisfy the Kraft-Szilard nonequivalence:

$$\sum_{i=1}^{M} L^{-n_i} \leq 1$$

The code (instant and unique) can be more illustratively represented with a tree. The paths from the source node to the terminal node in consideration are the new code words. The code word is a sequence - concatenation of the node designations. For every code there is only a given number of nodes in the tree. The code-path from the source to the terminal node can not cross, of course, any of the finally determined nodes. The more probable words will be assigned shorter tree-branches and the less probable words longer ones. The branching of the graphic tree determines the tree order, and its height determines its degree. Scheme 1 depicts a tree of second order and 4th degree.



1 0
0 1 1
1 1 1
0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
1 1 0 0
1 1 0 1

Scheme 1. Graphic depiction of a tree for code I = 0128.

The binary code can be designated as well with an index I, which is a concatenation of decimal numbers $t_1, t_2, \ldots, t_M$, where $t_i$ is the number of words with a length of i symbols. For example, index I = 104 defines a series of 5 code words, where the length of one word is 1 and three symbols constitute four words. Scheme 1 shows the vocabulary and the graphic depiction of a code index I = 0128.

## ALGORITHM DESIGN

The finite series of symbols X is at our disposal at the data source output. The statistical probability P assigned to them is also known. The entire finite-alphabet symbol series X is read into the vector X, for example, and the statistical probability assigned to each symbol is read into vector P. The X- and P-vector lengths are clearly the same and indices i coincide. For this reason, vector P with its arrangement of statistical probabilities suffices for further manipulations. The P-vector length must be increased to 2M - 1 due to the processing and storing of intermediate data.

The added vector components are at first filled with trivial data (eg., probability value >1). Using vectors X and P, the desired code table is formed, and it is used also to perform the coding process.

For example, for $X = (x_1, x_2, x_3)$ and $P = (p_1, p_2, p_3)$ the vector length M = 3. The needed space for the vector is 2M - 1 = 5 dimensional vector. Let us insert into vector P randomly selected values for $p_1 = 0.5$; $p_2 = 0.3$; $p_3 = 0.2$, and the other vector terms are completed with trivial values 2.0. Every vector component carries an index i, i = 1,2,..., 2M-1.

| index (i) | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| vector | 0.5 | 0.3 | 0.2 | 2.0 | 2.0 |

Proper statistical probabilities are not necessary in the first half of the vector P.

Terms with lowest values are chosen in pairs from vector P. In our example these are the terms with indices 3 and 2. A new multidimensional vector or a desired code table CODE (J,K) is formed with the chosen indices. The value of the index J in the code table CODE is determined by the finite number of symbols that constitute the code words. For binary code notation, J = 2.

Index K in the table formed is M - 1. Index with the lower probability value is placed in the first row of the code table, the other in the second row. Binary symbol 1 is assigned to all the code table indices in the first row and the symbol 0 to those in the second row.

In our example, the code table at first appears as

|  | Column 1 |  |
|---|---|---|
| Line 1 | 3 | assigned binary value 1 |
| Line 2 | 2 | assigned binary value 0 |

and with every selection of P-vector terms and code-table formation the table increases by an additional column.

Both values of the selected P-vector terms are added (0.5) and this value replaces the closest P-vector component with a trivial value (term with index 4). The terms in the first half of the vector that have already been used, are likewise replaced with trivial probability values. Thus we avoid re-using the terms that have already been processed (terms with indices 3 and 2). The new vector is now

| index (i) | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| vector | 0.5 | 2.0 | 2.0 | 0.5 | 2.0 |

The process is repeated until the smallest vector term exceeds the value of 1, and that at the same time signals that we have entirely exhausted the needed list of symbols.

When finished, the procedure presents us with one of the possible code tables that is characteristic for a given probability distribution.

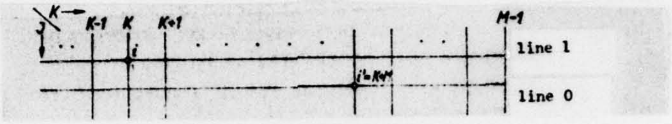
Due to the low memory-bank requirement such a code table is very useful for coding and decoding.

The final code table CODE in our example appears as

|  | Column 1 | Column 2 |  |
|---|---|---|---|
| Line 1 | 3 | 1 | assigned binary values 1 |
| Line 2 | 2 | 4 | assigned binary values 0 |

With a large number of symbols and with different statistical distribution, the formation of the code table CODE (2,M-1) is most facile with the use of a computer.

Code table CODE (2,M-1) and vector X are used for coding. In the code table CODE (2,M-1) one finds the index i for the symbol to be coded; the index appears at a random place in the table in column K and line J (symbol coordinates - indices). See Scheme 2.



Scheme 2. Index manipulation in a random code table.

In the next operation, our finite number of code symbols M is added to the column index K, where the desired index was located. The value of the new index is now i' = K + M. The change of index value by columns is accompanied by the change by code-table lines. The line index-location is designated by code symbols. When repeating the process, those (in our case binary) line-values are connected - concatenated that contained the index i. The process is terminated when an individual index in the code table exceeds the value 2M - 2.

The concatenated assembly of code symbols (binary patterns) is turned around and that is the desired code word.

Let us illustrate this with our code table. If we wish to code, for example, the input symbol defined by the statistical probability p = 0.2 and characterized by the index 3, then the procedure described above gives us for this symbol the code word 01, for the symbol with index 2 code word 00, and for the symbol with index 1 code word 1.

| Symbol | Probability | Code word |
|---|---|---|
| $x_1$ | $p_1 = 0.5$ | 1 |
| $x_2$ | $p_2 = 0.3$ | 00 |
| $x_3$ | $p_3 = 0.2$ | 01 |

The code table CODE (J,M-1) and vector X are used for decoding as well; the procedure is the reverse of coding.

One starts with the first code symbol which defines the code-table line and with the last code-table column. These two coordinates determine index i in the last code-table column. The new index is obtained by subtracting from the previous index the finite number of symbols M giving a new code-table column. The code signal that arrives next, defines the line, and thus the new index location is obtained.

The procedure is repeated until the code-table index-value becomes $\leq$ M. The last code-table number i is at the same time also the index of the desired symbol $x_i$.

In our example, if the code word 00 is to be decoded, we begin with index i in the second line of the second column. The new column value is obtained by K = i - M, and the second code-word symbol designates the symbol in the wanted line, ie., $x_2$.

All the necessary vectors and code table CODE can be stored in a memory that does not require a lot of space, and the times of coding and decoding are short.

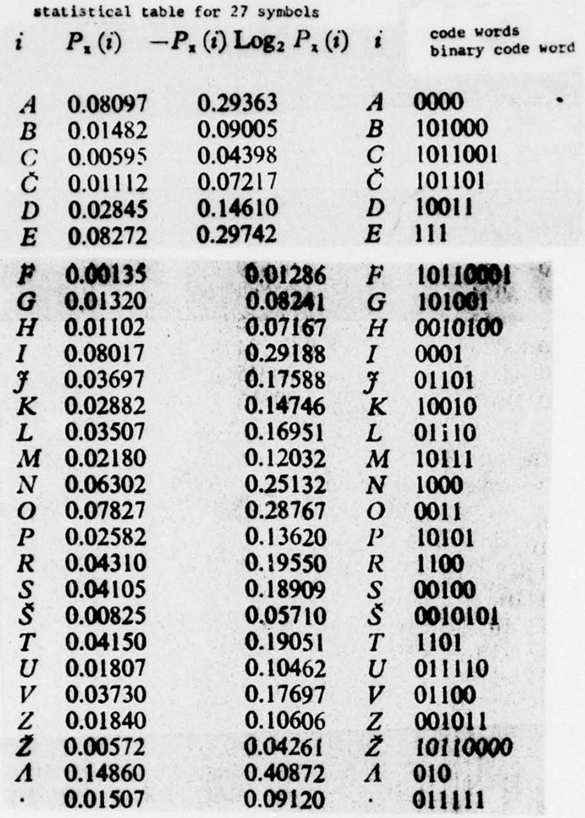
The memory, however, may contain only the code table whereas the needed vectors are restricted to the transmitting and receiving station to protect the messages.

Let us look at two illustrative examples. The first one deals with the coding of Slovenian literary text into code words consisting of two different symbols during the message transmission. The data source output consists of data corresponding to the 25 letters of the Slovenian alphabet, one symbol for denoting spaces, and one for all the punctuations - a total of 27 symbols.

The alphabet is

$$\mathscr{A} = \{A, B, C, D, \ldots, U, V, Z, \check{Z}, \Lambda, .\}$$

The symbols are independent of each other and unevenly probable. All the symbols are assigned a statistical probability of appearance frequency in the text[2].



statistical table for 27 symbols

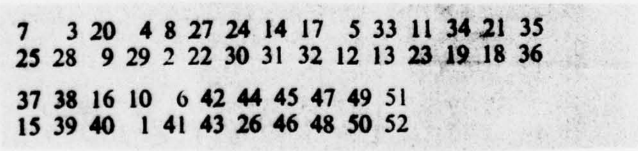
| *i* | $P_x(i)$ | $-P_x(i)\,\mathrm{Log}_2\,P_x(i)$ | *i* | code words<br>binary code word |
|---|---|---|---|---|
| *A* | 0.08097 | 0.29363 | *A* | 0000 |
| *B* | 0.01482 | 0.09005 | *B* | 101000 |
| *C* | 0.00595 | 0.04398 | *C* | 1011001 |
| *Č* | 0.01112 | 0.07217 | *Č* | 101101 |
| *D* | 0.02845 | 0.14610 | *D* | 10011 |
| *E* | 0.08272 | 0.29742 | *E* | 111 |
| *F* | 0.00135 | 0.01286 | *F* | 10110001 |
| *G* | 0.01320 | 0.08241 | *G* | 101001 |
| *H* | 0.01102 | 0.07167 | *H* | 0010100 |
| *I* | 0.08017 | 0.29188 | *I* | 0001 |
| *J* | 0.03697 | 0.17588 | *J* | 01101 |
| *K* | 0.02882 | 0.14746 | *K* | 10010 |
| *L* | 0.03507 | 0.16951 | *L* | 01i10 |
| *M* | 0.02180 | 0.12032 | *M* | 10111 |
| *N* | 0.06302 | 0.25132 | *N* | 1000 |
| *O* | 0.07827 | 0.28767 | *O* | 0011 |
| *P* | 0.02582 | 0.13620 | *P* | 10101 |
| *R* | 0.04310 | 0.19550 | *R* | 1100 |
| *S* | 0.04105 | 0.18909 | *S* | 00100 |
| *Š* | 0.00825 | 0.05710 | *Š* | 0010101 |
| *T* | 0.04150 | 0.19051 | *T* | 1101 |
| *U* | 0.01807 | 0.10462 | *U* | 011110 |
| *V* | 0.03730 | 0.17697 | *V* | 01100 |
| *Z* | 0.01840 | 0.10606 | *Z* | 001011 |
| *Ž* | 0.00572 | 0.04261 | *Ž* | 10110000 |
| *Λ* | 0.14860 | 0.40872 | *Λ* | 010 |
| · | 0.01507 | 0.09120 | · | 011111 |

Λ is the symbol for space

. is the symbol for all punctuations

The median data value for individual symbol is i(x) = 4.25304 bits/symbol.

Code table CODE:



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 3 | 20 | 4 | 8 | 27 | 24 | 14 | 17 | 5 | 33 | 11 | 34 | 21 | 35 |
| 25 | 28 | 9 | 29 | 2 | 22 | 30 | 31 | 32 | 12 | 13 | 23 | 19 | 18 | 36 |
| 37 | 38 | 16 | 10 | 6 | 42 | 44 | 45 | 47 | 49 | 51 | | | | |
| 15 | 39 | 40 | 1 | 41 | 43 | 26 | 46 | 48 | 50 | 52 | | | | |

If the code-word vocabulary were to be represented grafically, a tree of 8th degree and 2nd order would result. The minimal-redundancy codes, constructed mainly for economic reasons, would be even more successful if the interdependence of individual symbols were considered.

As a second example let us choose a data system with 16 elementary finite states defined by the series

$$\mathcal{X} = \{\alpha, \beta, \gamma, \dots, 0, \pi\}$$

All the states are taken to be independent of each other, and their appearance is unevenly probable. In this case, due to the chosen and assigned statistics, the graphical picture of the vocabulary appears as a one-sided tree.

statistical table 16 symbols

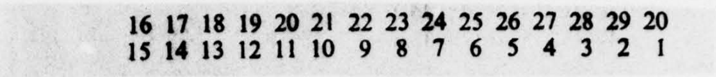
| $i$ | $P_x(i)$ | $-P_x(i)\,\mathrm{Log}_2\,P_x(i)$ |
|---|---|---|
| $\alpha$ | 0.50000 | 0.50000 |
| $\beta$ | 0.25000 | 0.50000 |
| $\gamma$ | 0.12500 | 0.37500 |
| $\delta$ | 0.06250 | 0.25000 |
| $\varepsilon$ | 0.03125 | 0.15625 |
| $\zeta$ | 0.01562 | 0.09372 |
| $\eta$ | 0.00781 | 0.05467 |
| $\vartheta$ | 0.00390 | 0.03120 |
| $\iota$ | 0.00195 | 0.01755 |
| $\varkappa$ | 0.00097 | 0.00970 |
| $\lambda$ | 0.00048 | 0.00529 |
| $\mu$ | 0.00024 | 0.00288 |
| $\nu$ | 0.00012 | 0.00156 |
| $\xi$ | 0.00006 | 0.00084 |
| $o$ | 0.00003 | 0.00045 |
| $\pi$ | 0.00001 | 0.00016 |

code words

| i | binary code word |
|---|---|
| $\alpha$ | 0 |
| $\beta$ | 10 |
| $\gamma$ | 110 |
| $\delta$ | 1110 |
| $\varepsilon$ | 11110 |
| $\zeta$ | 111110 |
| $\eta$ | 1111110 |
| $\vartheta$ | 11111110 |
| $\iota$ | 111111110 |
| $\varkappa$ | 1111111110 |
| $\lambda$ | 11111111110 |
| $\mu$ | 111111111110 |
| $\nu$ | 1111111111110 |
| $\xi$ | 11111111111110 |
| $o$ | 111111111111110 |
| $\pi$ | 111111111111111 |

The median data value for individual symbol is i(x) = 1.99932 bits/symbol.

Code table CODE:



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 20 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

The median data value for individual symbol is less favorable in the second example due to the highly uneven statistical probability. The information quantity can be improved by an appropriate choice of output states. The message information contents can also be improved by considering state interdependence.

CONCLUSION

The described coding method has justified its hopes. A small memory space is needed and the processing times are short. The choice of code symbols in the code table determines the synchronizing properties of the code. Random code tables can be used for secret message transmission. All sorts of code tables are best handled by a computer.

It is well known that coding into code words of variable length is characterized by incomplete reliability of reception. Messages will be properly received from the data source only when the receiving end has enough time for immediate decoding. If that is not the case, we can be sure that the receiver will not be able to receive the transmitted data in their entirety. Intermediaries are required that are built into the transmitting and receiving end. Due to the short processing times, the described coding method requires intermediaries of lower capacity.

LITERATURE

1. Huffman, D.A., A method for construction of minimum-redundancy codes, Proc. IRE, Vol. 40, September 1952.
2. Gyergyek, L., Prispevek k statistični obdelavi slovenskega pisanega besedila, Elektrotehniški vestnik (A contribution to the statistical treatment of Slovenian written text, Journal of electrical technology), 40, 1973, Ljubljana.

3. Rudner, B., Construction of minimum-redundancy codes with an optimum synchronizing property, IEEE Trans. on Inf. Theory, Vol. IT-17, No. 4, July 1971.

AUTHOR'S ADDRESS

Bogomir Horvat, M.S., graduated engineer, professor of higher education at Higher technical school, Maribor, Smetanova 17.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER
FTD-ID(RS)T-1312-77

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*
CONSTRUCTION OF MINIMAL-REDUNDANCY CODES FOR VARIABLE-LENGTH WORDS

5. TYPE OF REPORT & PERIOD COVERED
Translation

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR*(s)*
B. Horvat

8. CONTRACT OR GRANT NUMBER*(s)*

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Foreign Technology Division
Air Force Systems Command
U. S. Air Force

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

12. REPORT DATE
1976

13. NUMBER OF PAGES
10

14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*

15. SECURITY CLASS. *(of this report)*
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

09

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*